CLAIMS

What is claimed is:

1. In a client-server environment, a method of providing a graphical user interface (GUI) to an end-user, the method comprising:

a client application receiving commands from a server application, the commands dictating a GUI implementation to be displayed to an end-user, the GUI implementation revealed to the client application only at run time; and

the client application returning events to the server application, the events indicating state change in the client application.

2. The method of claim 1, wherein the server application and client application communicate with each other without being bound through linking.

3. The method of claim 1, further comprising:

the client application displaying to the end-user a list of server applications with logon details;

the client application accepting from the end-user a selection from the list; and

the client application presenting a minimal set of interface elements to allow the end-user to communicate with the server application.

4. The method of claim 1, wherein the client application connects to a default server application.

5. The method of claim 1, further comprising:

the client application receiving resources from the server application, the received resources having been separately compiled into binary form and stored in a dynamic linked library for runtime efficiency, the dynamic linked library moved independently of the client application; and

the client application reserving binding for runtime by caching the dynamic linked library.

6. The method of claim 1, further comprising:

the client application receiving resources from the server application, the received resources being defined using human readable textual descriptions, the resources moved independently of the client application;

the client application reserving binding for runtime by caching the resources;

and

the client application processing the resources into runtime form.

7. The method of claim 1, wherein the client application is based on a protocol that eliminates a requirement for HTML as a method of presenting the GUI to the end-user.

8. The method of claim 1, wherein the GUI implementation is displayed and utilized based on a protocol, the protocol eliminating a requirement for producing any custom logic in the client application.

9. The method of claim 1, wherein the client application communicates with the server application using a protocol, the protocol reducing the number of API calls made between the client application and server application.

10. The method of claim 1, further comprising the client application allowing the end-user to manipulate the list of server applications.

11. The method of claim 1, wherein the server application executes on a different platform than the client application.

12. The method of claim 1, wherein the server application executes on the same platform as the client application.

13. The method of claim 1, wherein the client application can connect simultaneously to a plurality of server applications.

14. A computer-readable medium containing instructions which, when executed by a computer, provide a graphical user interface (GUI) to an end-user, by:

directing a client application to receive commands from a server application, the commands dictating a GUI implementation to be displayed to an end-user, the GUI implementation revealed to the client application only at run time; and

directing the client application to return events to the server application, the events indicating state change in the client application.

15. The method of claim 1, further comprising instructions for directing the client application to communicate with the server application without being bound to the server application through linking.

16. The computer-readable medium of claim 14, further comprising instructions for:

directing the client application to display to the end-user a list of server applications with logon details;

directing the client application to accept from the end-user a selection from the list; and

directing the client application to present a minimal set of interface elements to allow the end-user to communicate with the server application.

17. The computer-readable medium of claim 14, further comprising instructions that direct the client application to connect to a default server application.

18. The computer-readable medium of claim 14, further comprising instructions that for:

directing the client application to receiv resources from the server application, the received resources having been separately compiled into binary form and stored in a dynamic linked library for runtime efficiency, the dynamic linked library moved independently of the client application; and

directing the client application to reserve binding for runtime by caching the dynamic linked library.

19. The computer-readable medium of claim 14, further comprising instructions for:

directing the client application to receive resources from the server application, the received resources being defined using human readable textual descriptions, the resources moved independently of the client application;

directing the client application to reserve binding for runtime by caching the resources;

and

directing the client application to process the resources into runtime form.

20. The computer-readable medium of claim 14, further comprising instructions for directing the client application to comply with a protocol that eliminates a requirement for HTML as a method of presenting the GUI to the end-user.

21. The computer-readable medium of claim 14, further comprising instructions for directing the GUI implementation to be displayed and utilized based on a protocol, the protocol eliminating a requirement for producing any custom logic in the client application.

22. The computer-readable medium of claim 14, further comprising instructions for directing the client application to communicate with the server application using a protocol, the protocol reducing the number of API calls made between the client application and server application.

23. The computer-readable medium of claim 14, further comprising instructions for directing the client application to allow the end-user to manipulate a list of server applications.

24. The computer-readable medium of claim 14, further comprising instructions for directing the client application to execute on a different platform than the server application.

25. The computer-readable medium of claim 14, further comprising instructions for directing the client application to execute on the same platform as the server application.

26. The computer-readable medium of claim 14, further comprising instructions for directing the client application to connect simultaneously to a plurality of server applications.

27. A client application for use in a client-server environment, the client application comprising:

means for receiving commands from a server application, the commands dictating a GUI implementation to be displayed to an end-user, the GUI implementation revealed to the client application only at run time; and

means for returning events to the server application, the events indicating state change in the client application.

28. The client application of claim 27, further comprising means for the server application and client application to communicate with each other without being bound through linking.

29. The client application of claim 27, further comprising:

means for displaying to the end-user a list of server applications with logon details;

means for accepting from the end-user a selection from the list; and

means for presenting a minimal set of interface elements to allow the end-user to communicate with the server application.

30. The client application of claim 27, further comprising means for connecting to a default server application.

31. The client application of claim 27, further comprising:

means for receiving resources from the server application, the received resources having been separately compiled into binary form and stored in a

dynamic linked library for runtime efficiency, the dynamic linked library moved independently of the client application; and

means for reserving binding for runtime by caching the dynamic linked library.

32. The client application of claim 27, further comprising:

means for receiving resources from the server application, the received resources being defined using human readable textual descriptions, the resources moved independently of the client application;

means for reserving binding for runtime by caching the resources;

and

means for processing the resources into runtime form.

33. The client application of claim 27, wherein the client application is based on a protocol that eliminates a requirement for HTML as a method of presenting the GUI to the end-user.

34. The client application of claim 27, wherein the GUI implementation is displayed and utilized based on a protocol, the protocol eliminating a requirement for producing any custom logic in the client application.

35. The client application of claim 27, wherein the client application communicates with the server application using a protocol, the protocol

reducing the number of API calls made between the client application and server application.

36. The client application of claim 27, further comprising means for allowing the end-user to manipulate the list of server applications.

37. The client application of claim 27, wherein the server application executes on a different platform than the client application.

38. The client application of claim 27, wherein the server application executes on the same platform as the client application.

39. The client application of claim 27, further comprising means for connecting simultaneously to a plurality of server applications.